CONOSCERE IL COMPUTER DIRETTAMENTE DAL COMPUTER

per Commodore Vic20 e 64

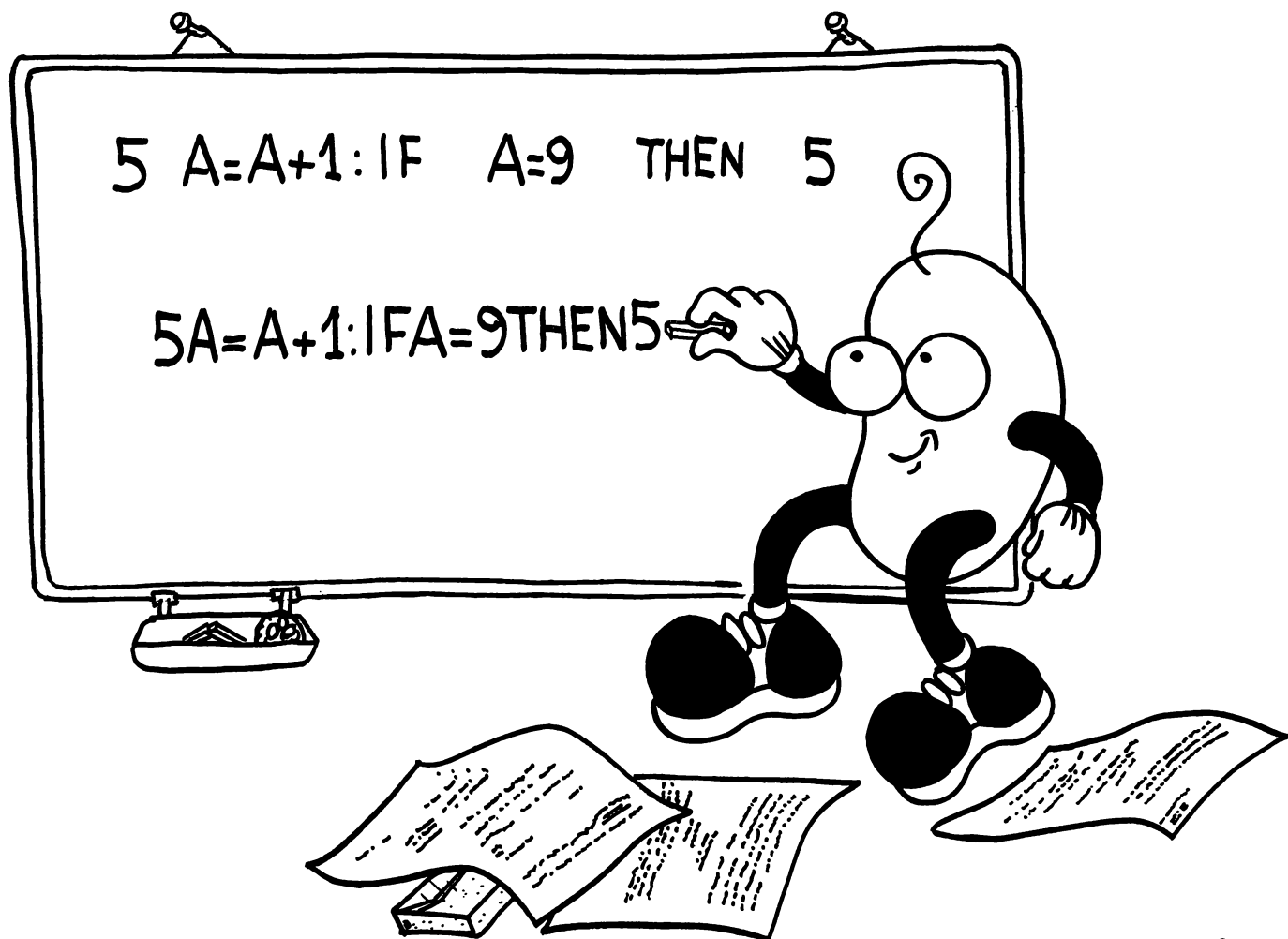
Per creare un buon programma non basta imparare come usare le istruzioni, anche se questo è sicuramente indispensabile, ma bisogna anche sapere impostare (e quindi capire) la logica del programma che si intende costruire. Per questo l'argomento di questa lezione, cioè **L'OTTIMIZZAZIONE DEI PROGRAMMI**, è particolarmente importante.

Infatti per ottimizzazione si intende la capacità di rendere i programmi più chiari, più veloci in esecuzione e che occupino meno memoria, evitando la scrittura di listati tortuosi (ad esempio con istruzioni o salti che potrebbero anche essere eliminati).

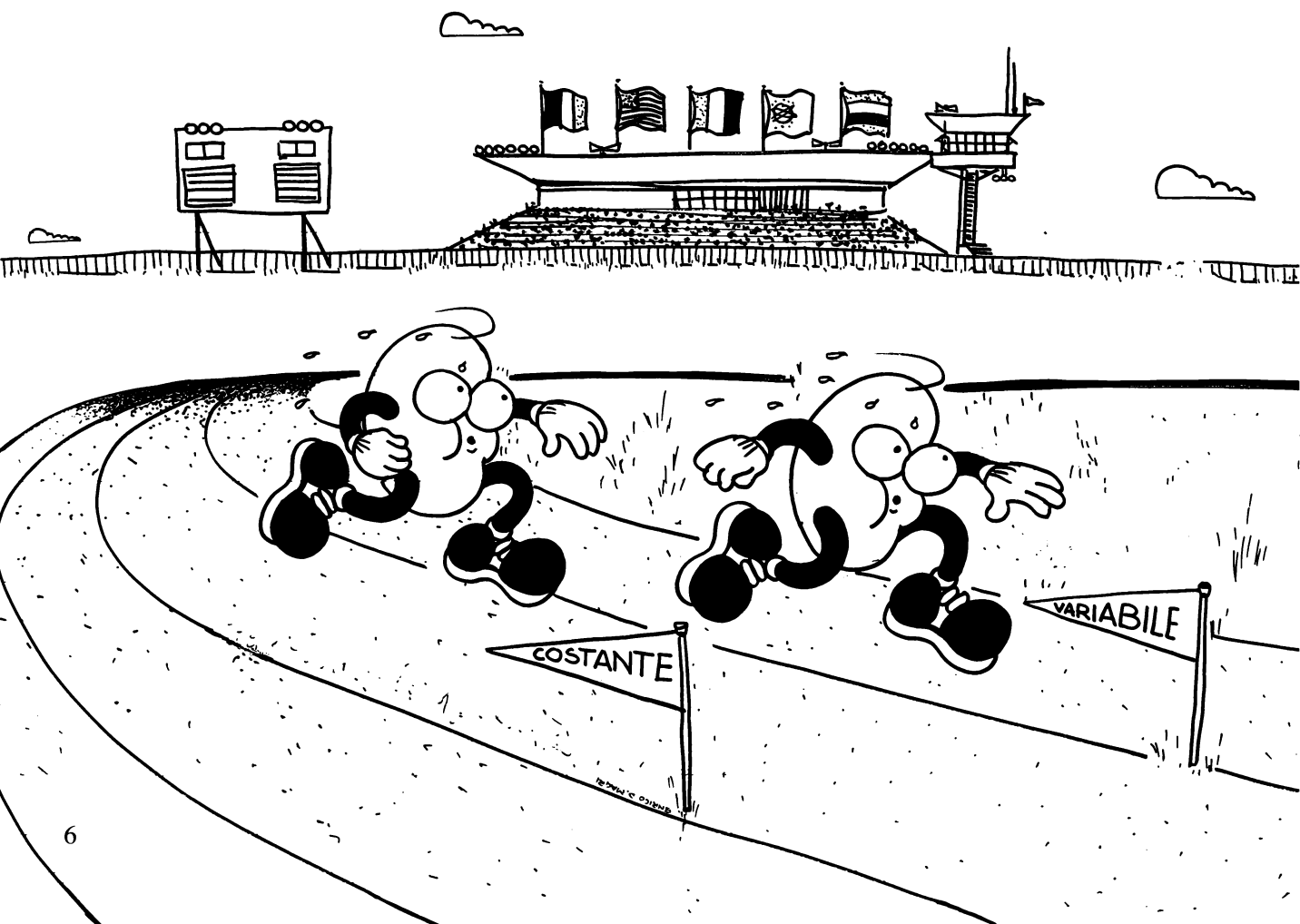
Quando accendi il computer hai a disposizione 38911 bytes di memoria libera.

Per conoscere, in qualsiasi momento, quanta memoria libera ti resta scrivi l'istruzione **PRINT FRE (Ø)**.

In questo modo ti verrà mostrato il numero dei bytes ancora liberi.

Tuttavia, a volte, potrai ottenere in risposta dei valori negativi e quindi apparentemente assurdi. In questo caso basterà aggiungere al valore ottenuto il numero 65536. Quindi, ad esempio, se con **PRINT FRE (Ø)** il risultato è **-26627**, i bytes liberi saranno in realtà **38908** (cioè: **-26627** + **65536**).
Tieni presente che durante l'esecuzione del programma la quantità di memoria libera si ridurrà ancora, visto che verrà usata per memorizzare i contenuti delle variabili. Quindi se vorrai conoscere la memoria occupata dal solo programma, dovrai prima azzerare le variabili con il **CLR** (lez. 16) e poi eseguire il **PRINT FRE (Ø)**.
Ma vediamo, a questo punto, alcuni importanti consigli da ricordare durante la stesura dei tuoi programmi.
1) Ricorda che se scrivi le istruzioni senza separarle con gli spazi risparmierai memoria.

2) Quando è possibile, cioè quando conosci il numero dei giri, usa sempre cicli con il FOR NEXT anzichè realizzarli con l'IF e un contatore. In questo modo il programma in esecuzione risulterà più veloce.
3) Sulla stessa linea di programma cerca di mettere più istruzioni che nell'insieme svolgono la stessa funzione.
4) Cerca di limitare al minimo il numero di variabili, vettori e matrici all'interno del programma. Quindi, se ti è possibile, usa la stessa variabile corrente nei cicli FOR NEXT.
Comunque in caso di necessità puoi dichiarare le variabili, i vettori e le matrici come numeri interi, aggiungendo il segno **%** dopo il nome.
Tieni presente che nelle variabili, nei vettori e nelle matrici dichiarate intere, potrai usare solo i numeri compresi tra **-32768** e **32767**.
5) Dopo aver creato gli INPUT ricordati di controllare se le risposte inserite sono accettabili. In caso contrario fai ripetere l'inserimento.
6) Non usare i commenti nelle istruzioni REM in modo eccessivo. E' sufficiente metterne uno solo all'inizio di ogni suddivisione di programma.

7) Limita al minimo i salti con il GOTO, in modo da non creare disordine e rendere così più facile il controllo del listato. Evita anche salti a linee che contengono un GOTO ad un'altra linea.

8) Se il programma contiene subroutine accodale al programma principale.
Quindi non creare inutili subroutine che verrebbero richiamate da un solo punto del programma.
Tieni presente che, specialmente in programmi molto lunghi, l'esecuzione delle ultime linee è più lenta delle prime. In questo caso ti converrà mettere le parti di programma più lente, o le subroutine di uso più frequente, all'inizio del programma.
9) Usa sempre, specialmente in cicli molto lunghi, varabili e non costanti, questo per velocizzare l'esecuzione.

Listato dell'esercizio: ESEMPIO PRATICO

```
100 dimn(300)
110 i=ti
120 K=K+1
140 r=int(rnd(0)*1000+1)
150 n(K)=r
160 ifK<300then120
200 forK=1to300
210 t=t+n(K)
220 printn(K)
230 nextK
300 print"totale";t
320 m=int(t/300)
330 print"media";m
340 w=int((ti-i)/60)
350 print"secondi impiegati";w
360 end
```

Listato dell'esercizio: COMPLETA IL LISTATO

```
10 forK=1to[*]:readn$(K),i$(K):nextK
20 for[*]to10
30 [*]=int(rnd(0)*3+1)
40 w=[*](rnd(0)*3+1)
50 [*]=n$(r):n$(r)=n$(w):n$(w)=c$
60 c$=i$(r):i$(r)=i$([*]):i$(w)=c$
70 next[*]
80 [*]K=1to3
90 print"chi ha inventato il:"
100 print[*]
110 input"inventore";[*]
120 ifs$=i$(K)thenprint"esatto":goto[*]
130 print"sbagliato, era":printi$(K)
140 [*]K
150 data"telefono","meucci"
160 data"parafulmini","franklin"
170 data"cannocchiale","galilei"
```

PROGRAMMIAMO INSIEME (CBM 64)

```
10 poke53280,7:poke53281,7
20 dims$(13)
30 c$(1)="1":c$(2)="x":c$(3)="2"
40 d$(1)="1x":d$(2)="x2":d$(3)="12"
50 t$(1)="1x2"
60 input"numero di triple";t
70 ift>13then60
80 input"numero di doppie";d
90 ifd>13-tthen80
100 n=int((3↑t)*(2↑d))
110 print"n. colonne:";n
120 l=n*350
130 print"costo l.";l
140 ift=0then180
150 fork=1tot
160 gosub300:s$(r)=t$(1)
170 nextk
180 ifd=0then220
190 fork=1tod
200 gosub300:q=int(rnd(0)*3+1):s$(r)=d$(q)
210 nextk
220 ifd+t=13then260
230 fork=1to13-d-t
240 gosub300:q=int(rnd(0)*3+1):s$(r)=c$(q)
250 nextk
260 print:fork=1to13
270 print"partita n.";k;tab(15);s$(k)
280 nextk
290 end
300 r=int(rnd(0)*13+1)
310 ifs$(r)<>""then300
320 return
```

PROGRAMMIAMO INSIEME (VIC 20)

Il listato è identico a quello per il CBM 64 tranne nella linea:

```
10 poke 36879,127
```

Soluzione dell'esercizio: COMPLETA IL LISTATO (lez. 23)

```
10 printchr$(147):a$(1)="N":a$(2)="M"
20 forx=1to7:fork=1to5
30 t(1)=ti
40 c$=a$(int(rnd(0)*2+1))
50 d$=a$(int(rnd(0)*2+1))
60 print" "c$" "d$;
70 if(e$=a$(1))*(c$=a$(2))thens=s+1
80 e$=d$
90 if(c$=a$(1))*(d$=a$(2))thens=s+1
100 nextk:print:print:nextx:print
110 input"sequenze N M";p
120 t(2)=ti
130 ifp<>sthenprint"sbagliato":end
140 print"esatto"
150 i=int((t(2)-t(1))/60)
160 print"secondi impiegati:";i
```